Multi-Standard Infrared Receiver

compatible with (almost) all remote control transmitters

This circuit complements the remote controls with which practically all consumer electronics devices are equipped: a receiver which can operate with a wide range of infrared transmitters.

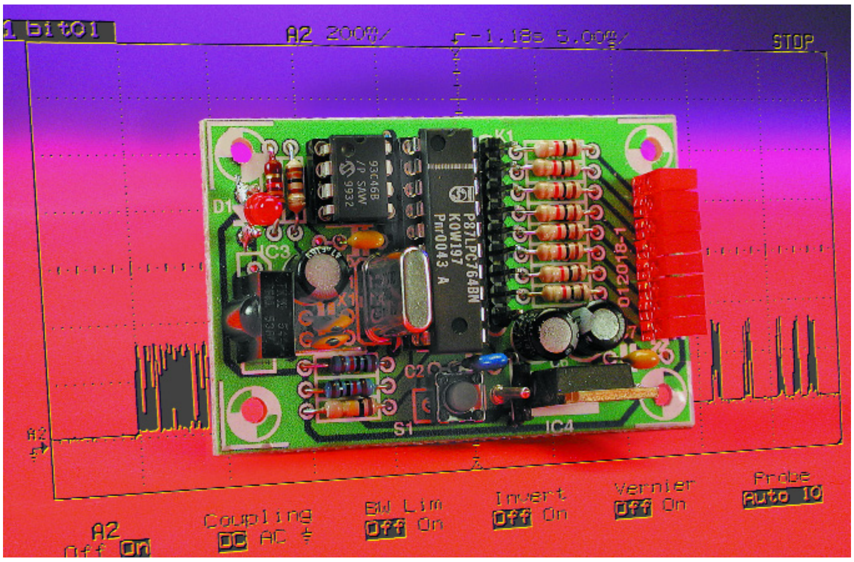
Given the information on remote controls that appeared in the March and April 2001 issues of *Elektor Electronics*, the final remaining step is to build a genuinely universal multi-standard infrared receiver. Using this, a remote controlled system can be constructed using practically any readily-available transmitter, and remote controls that would otherwise be scrapped can be given a new lease of life. Also, the infrequently-used buttons (such as the automatic programming buttons on a TV remote control), can have a new function and be used for example to control the room lighting. Unfortunately, remote controls that transmit using 'flash' mode cannot be used, owing to the way the infrared receiver works internally.

The block diagram of the receiver is shown in **Figure 1**. Any desired key on the remote control can be used to operate the circuit: pressing the button causes the selected output to turn on or off. Built into a multi-way extension lead, up to eight channels could be controlled independently.

Model railway enthusiasts will appreciate the ability to easily control lights or other special functions using infrared light, avoiding messy wiring. You might even already have a suitable transmitter. If required, several receivers can be connected in parallel in order to decode all the functions available from a single transmitter. The number of code combinations is practically unlimited, allowing a vast range of remote control applications.

The circuit

The circuit in **Figure 2** is very simple, and should present no constructional difficulties. It is recommended to fit IC1 and IC2 in sockets.



Infrared reception

Receiver IC3 is a highly integrated device that can decode infrared signals with a set carrier frequency. It includes a highly sensitive infrared receiver which operates using a carrier frequency of 36 kHz. A photodiode with daylight filter, amplifier stages, filter and demodulator are all built in, so that no external circuitry worth mentioning is called for. R1 and C1 form a low-pass filter to eliminate supply-borne interference. Several alternatives for this IC are given in the parts list: be careful to note the pinouts!

LED D1 indicates operation of the circuit, flashing rhythmically on reception of data.

Microcontroller

The microcontroller has already been used in several constructional projects in *Elektor Electronics*, such as the Infrared Code Analyser in the October 2001 issue. It has the following characteristics:

- 4 kbytes ROM
- 128 bytes RAM
- 32 bytes customer code EEPROM
- 2.7 V to 6 V supply voltage
- two 16-bit counter/timers
- internal reset
- internal RC oscillator selectable
- 20 mA driver on all port pins
- maximum of 18 I/O pins, when internal reset and RC oscillator are selected.

Technical characteristics

- supply voltage: 5 V
- capable of learning the following infrared codes: 'Japanese', NEC, RC5, RECS80, SIRCS, Denon and Motorola, as well as 'far east' (as used by Daewoo for example)
- 8 freely programmable outputs
- all data stored in EEPROM
- optimised for 36 kHz carrier frequency
- visual programming confirmation
- visual indication of output states

- 2 analogue comparators
- I^2C interface
- full-duplex UART
- serial in-circuit programming

Using a clock frequency of 6 MHz and the internal divider (with a division ration of six) we have a cycle time of 1 ms. This allows accurate measurement of pulse widths, which is essential when deciding between the various codes. The pulse width is measured using one of the two internal timers and compared against preset threshold values.

EEPROM

The serial EEPROM IC2 has a storage capacity of 1024 bits, organised in a 64 by 16 matrix. All read and write operations take place over a Microwire-compatible interface in blocks of 16 bits. Once written, data are retained for at least 40 years, according to the manufacturer.

Switching stage

The microcontroller has drivers capable of delivering up to 20 mA per output pin, when switching to ground. At a high level only about 1 mA can be delivered, and so a transistor is required to switch the load. Two options are shown in the circuit diagram:

Option 1

The port pin drives the output stage via a BUZ11 MOSFET. With its R_{DSON} of 0.04 W, this transistor can easily handle continuous loads of 5 A. The diode protects the transistor against voltage spikes, if for example an inductive load is used. The diode can be dispensed with in the case of a resistive load. This option can only be used if direct currents are to be switched.

Option 2

In this variation a 12 V relay with a coil resistance of 400 W (that is, with a coil current of 30 mA) is switched via a BC548 transistor. The desired load can be switched using a small power relay. The flywheel diode is essential here.

Power supply

Voltage regulator IC4 produces the supply voltage of 5 V, as required by the receiver IC3. The input voltage to the regulator must be at least 9 V, since the regulator drops about 3 V. In applications where a 5 V supply is already available, IC4 can be dispensed with. Observe that cooling is required if the regulator is used at 1 A, but not at 100 mA or below. An input voltage of 12 V is recommended if 12 V relays are to be used.

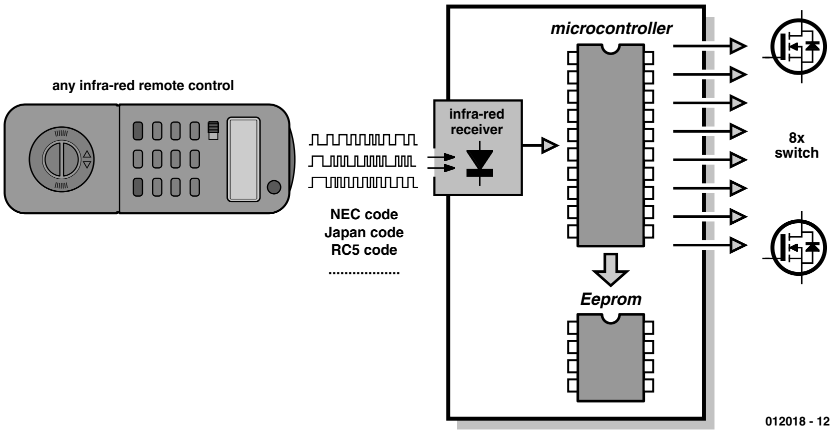


Figure 1. Block diagram of the multi-standard receiver

The software

The software is based on the Infrared Code Analyser, extended so that the received codes are stored and operate the output switches. Rather than simply sampling the signal and storing the samples, the actual code is read: for example the individual pulse lengths are compared against certain given reference values. This operation is rather less memory-intensive, but has the disadvantage that only known formats can be recognised. 'Learning' remote controls generally operate by simply storing samples of the signal and require a large static memory to store the quantity of data.

The software continuously samples the signal (rather more frequently than shown in **Figure 3**) by polling the logical state of input pin P1.4. This occurs at a rate determined by the software cycle time. During the polling process, a software counter counts the number of samples for which the input remains in each logical state. The count is a digital measure of the pulse length. Short interruptions or spikes in the signal can be ignored by only accepting a logic level when it has been stable for a certain minimum number of samples. If this minimum threshold is not reached, interference is assumed and the current counter value is discarded, counting continuing from the previous value.

As with all things, this procedure has its advantages and its disadvantages. On the one hand, it is easy to program, it is resilient (interference between samples is ignored when delay loops are used) and it is portable to other types of microcontroller, since no hardware resources are used. On the other hand, it places a great burden on the processor: the software is exclusively processing the samples and cannot get on with other tasks while a message is being received. Also, interrupts occurring during the measurement can lead to highly inaccurate readings, depending on the time it takes to service them.

The received codes are stored in hexadecimal form in the microcontroller's internal RAM and in the EEPROM. There is no distinction made between address and com-

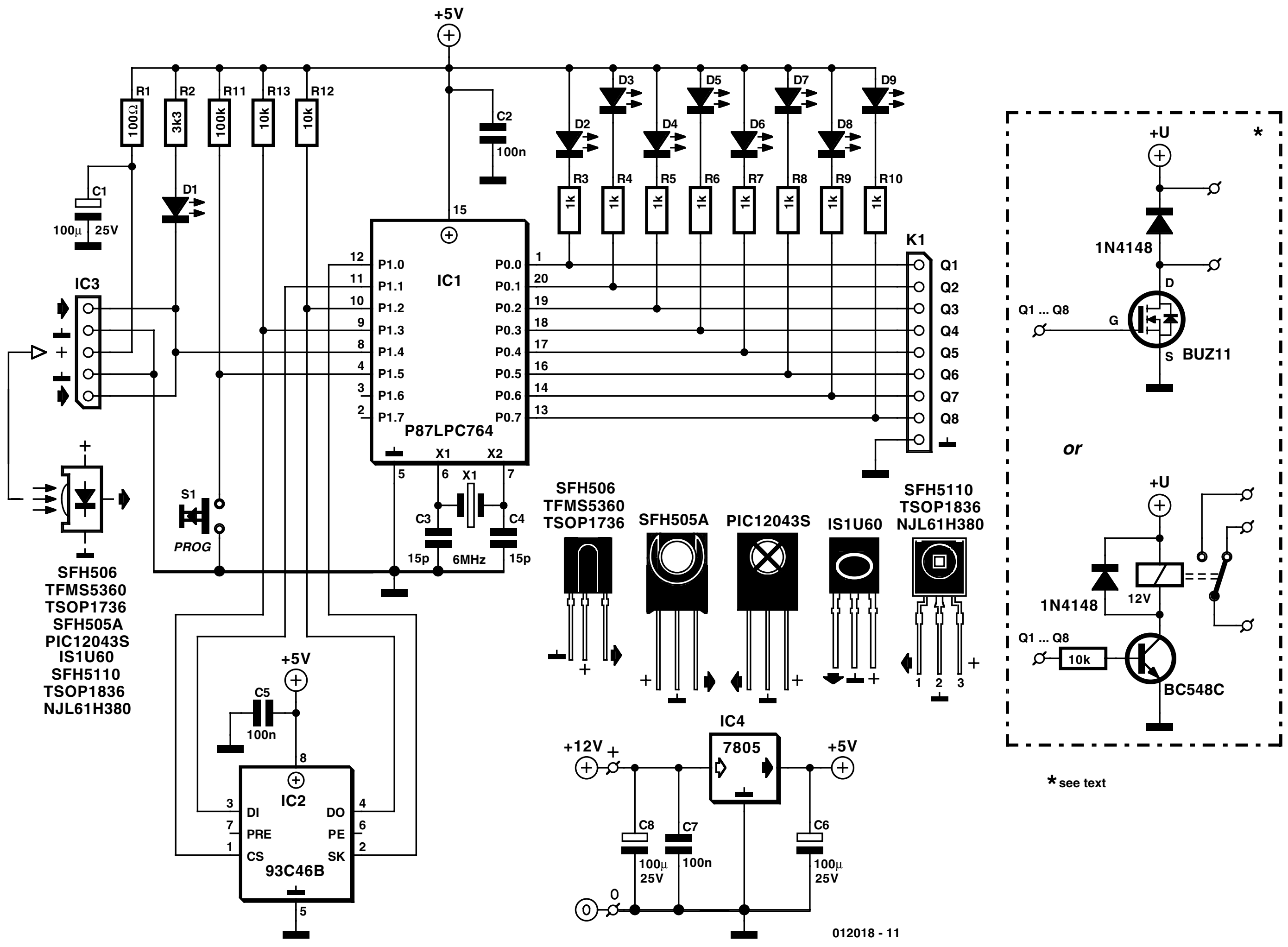


Figure 2. The circuit consists of just an infrared receiver module, a microcontroller, and an EEPROM

mand bits, and so the entire received message is compared with the stored one. 'Toggle' bits, such are used in the RC5 code, which change their state each time a button is pressed, are ignored by the software, since these would otherwise give rise to two different hexadecimal values for the code.

In use

Constructing the receiver circuit on the printed circuit board shown in **Figure 4** should present no problems. The pinouts of several suitable infrared receiver devices are shown.

Before the unit is installed it must be programmed, so that the microcontroller can learn which command is to be used to control which output. Programming is started by briefly pressing button S1. LED D2 immediately begins to flash. Each command now received using one of the codes understood by the microcontroller is stored and assigned to that port pin. This port pin is now programmed in, and the programming process for the next begins automatically: LED D3 starts to flash. This port pin can now be associated with a command in exactly the same way. It therefore does not matter if a different remote control transmitter, employing a different code, is used for each output.

The process continues until all eight outputs have been programmed, after which the LEDs stop flashing. If, at a later date, it is desired to change the command associated with just one channel,

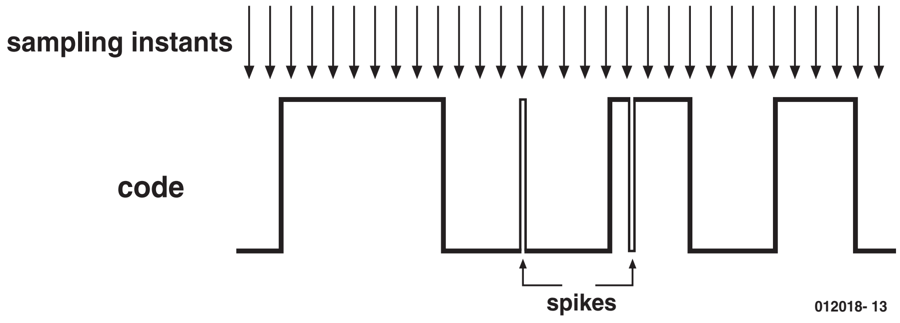


Figure 3. Sampling by polling

COMPONENTS LIST

Resistors:
R1 = 100Ω
R2 = 3kΩ3
R3-R10 = 1kΩ
R11 = 100kΩ
R12,R13 = 10kΩ

Capacitors:
C1,C2,C8 = 100µF 10V
C2,C5,C7 = 100nF ceramic
C3,C4 = 15pF

Semiconductors:
IC1 =P87LPC764BN (Philips), programmed, order code **012018-41**.
IC2 = 93C46 (Microchip 93C46B/P)
IC3 = TSOP1736 (alternatives: SFH5110-36, IS1U60, TFMS5360, PIC26043SM,TSOP1836)
IC4 = 7805
D1 = LED red, high efficiency
D2-D9 = LED, red, rectangular case

Miscellaneous:
X1 = 6 MHz quartz crystal
S1 = pushbutton (make contact)
20-way IC socket
8- way IC socket
K1 = 9-way SIL pinheader
Project software, on disk, order code **012018-11**, or free download from www.elektor-electronics.co.uk

the pins for which programming is not required can be skipped with a brief press of the button.

Note that while the LED is flashing, any connected piece of equipment will be switched on and off. All settings are stored permanently in EEPROM IC2, so that when the microcontroller is reset the most recently programmed configuration is still in force.

When a previously-stored code is received, the corresponding output changes state. Since many remote control transmitters send the same message repeatedly when a button is held down, a software monostable is provided to ensure clean operation. Only after a delay of around one second after the last valid message can the state of an output pin be changed again.

Because the software can only interpret the code formats it knows, the user can use the fact that the flashing LED advances as each output is programmed as confirmation that the code has been correctly read. If it is not read, the flashing LED will not advance.

Since we have used an infrared receiver IC3 optimised for a carrier frequency of 36 kHz, we can obtain the greatest range by employing a transmitter that operates on this frequency. If adequate range is not achieved, it is most likely because the transmitter uses a different carrier frequency.

(012018-1)

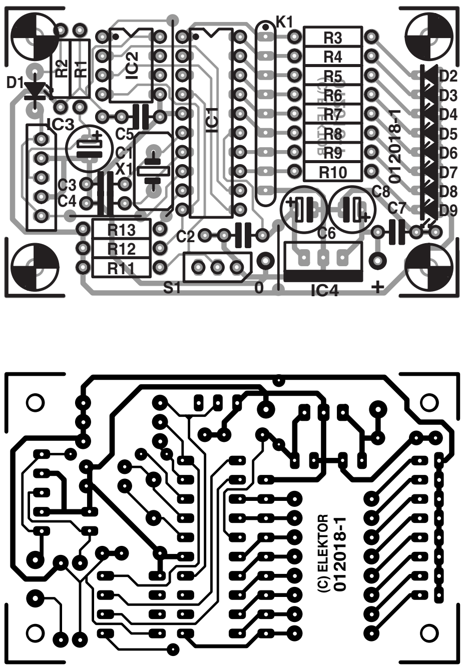


Figure 4. Printed circuit board layout.

Internet addresses

EEPROM:
www.fairchildsemi.com/pf/FM/FM93C46.html

Microcontroller:
www.semiconductors.philips.com/pip/p87lpc764bd

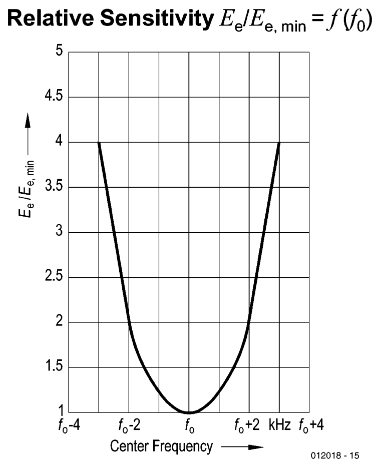
Infrared receiver IC:
www.infineon.com/cmc_upload/0/000/008/562/sfh5110.pdf
www.vishay.com/products/optoelectronics/IRMall.html

Carrier frequency

From **Figure A** we can clearly see that even with a transmitter that operates on 34 kHz or 38 kHz, we already have a loss of sensitivity by a factor of 2. If this presents a problem, it may help to use a different infrared receiver. The ICs are available for practically all centre frequencies in 2 kHz steps from 30 kHz to 40 kHz.

A simple way to determine the transmitter carrier frequency is to use an oscilloscope. An ordinary infrared photodiode such as a BPW75 can be connected to 5 V via a resistor as shown in **Figure B**. If a remote control is held close to the photodiode, the signal on the oscilloscope can be analysed and the carrier frequency measured.

A



B